A Geant4-based Beam Montecarlo?

- This follows the G4 presentation I gave in December at the analysis meeting (`http://www-boone.fnal.gov/software_and_analysis/meetings/index.html`),

- Then we agreed on having a "hands-on" experience to try to implement some G4 features which would be important for a G4-based beam MC

- What I am presenting today reprsents ∼2 months of my work, including time spent to learn some C++ and Root

- This would have taken much longer without the help of:

1. Eric Prebys (⇒ geometry implementation)
2. Bill Seligman, Nevis Labs (⇒ interface with Root)
3. Panagiotis (⇒ general G4 issues)

Outline

- MiniBooNE requirements from a beam MC
- Introduction to G4 (history, collaboration)
- General features, G4 classes description
- Results obtained so far
- “To-do” list

1. General G4 aspects are shown in blue
2. Specifics of what has been done for MiniBooNE in brown

MiniBooNE needs a flexible beam MC

- The business of hadronic production models is a complicated one:
  1. large differences (e.g. factors of 2) exist between models
  2. hard to say what is the best model today and two years from now
- Experimental input:
  1. whatever simulation model we will choose initially, it is very likely that we will change it based on some combination of Harp, E910, LMC data
  2. this experimental input will come in at various (and late) stages, and we need a clean an easy way to interface it to our beam MC

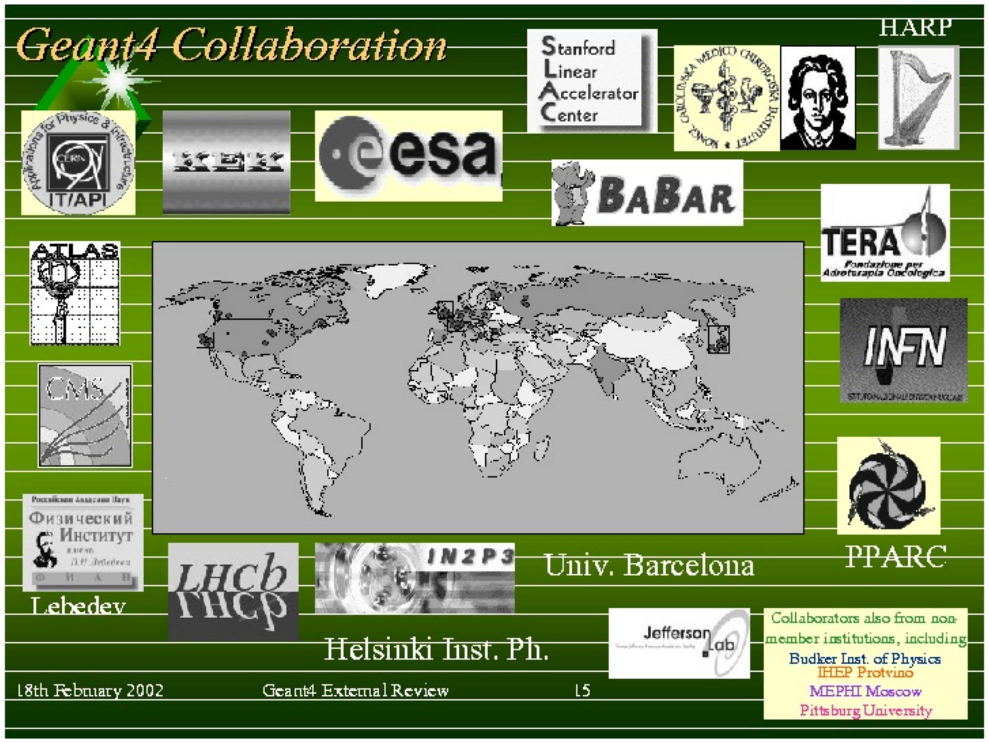
G4 History and Collaboration

History :

- Geant3 development and maintenance stopped in 1994; G4 project started
- First G4 production release in 1998
- Two scheduled public releases per year; last one in Dec ’01 (G4.4.0)

Collaboration :

- ∼150 physicists, none from FNAL, 7 from Harp



G4 code implemented by users

- G4 is written in C++, partitioned into loosely-coupled “classes”
- Classes $\sim$ collection of data members (sort of ntuples), plus “methods” defined on them (e.g. get/set functions to retrieve/set values of the data members)
- Documentation could be more detailed
- User code written so far: about 3000 lines
- Most of it copied from the very good set of template examples provided
- Same executable for interactive or batch simulation

User-defined classes

- Geometry construction: volumes, materials, em field

  `BooNEGeometryConstruction, BooNEField`

- Physics list: activate particles, physics processes, energy cuts

  `ExN04PhysicsList, ExN04EMPhysics, ExN04GeneralPhysics, ExN04HadronPhysics, ExN04IonPhysics, ExN04MuonPhysics`

- Primary generator action: define energy, position and angular distributions for primaries

  `BooNEPrimaryGeneratorAction`

- User actions (at run, event, stacking, tracking, step level)

  `BooNERunAction, BooNEEventAction, BooNETrajectory, BooNETrackingAction, BooNESteppingAction`

- Visualization interface: $\sim$ 10 different visualization drivers

  `ExN02VisManager`

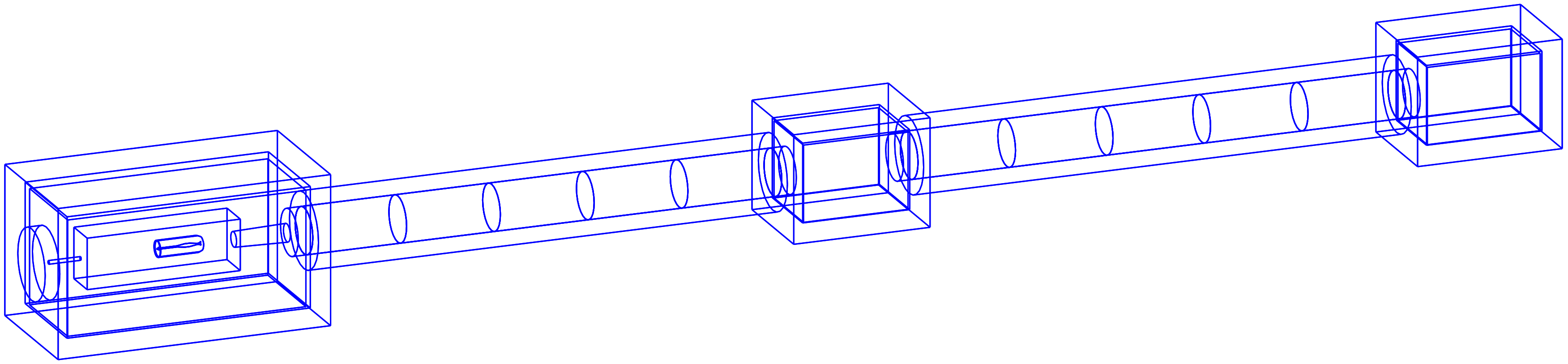
- Analysis interface: interface with analysis packages (e.g. Root)

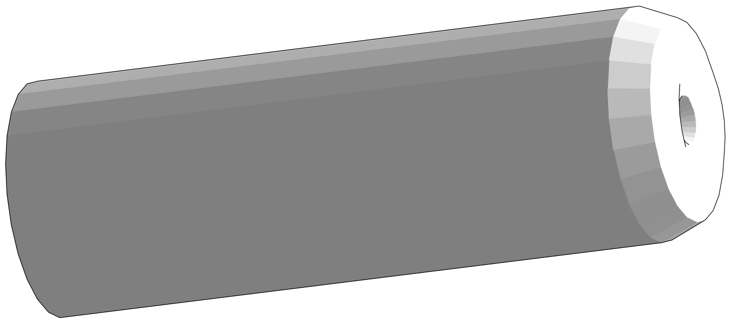
  `BooNEAnalysis`

Geometry construction

- Conceptually very similar to Geant3
- Capability of automatically converting materials and volumes specified in a G3 geometry file (`ugeom.F`) into a G4-readable format (`ugeom.dat`)
- This tool was used to convert the entire geometry and materials from the G3 beam MC, by Eric Prebys
- Capability of putting generic electromagnetic fields
  $\Rightarrow$ only rough schematization of horn field implemented so far:
  $1/r$ field for $0 < z < 180$ cm, $2.2 < r < 30$ cm
- There are better ways to implement the magnetic field locally; work is in progress

Geometry examples





Physics list

- There are no defaults: need to explicitly construct all particles and physics processes you are interested in
- Code looks like this (extracted from `ExN04HadronPhysics`):

```
void ExN04HadronPhysics::ConstructParticle() { ...
// Construct all barions
G4BaryonConstructor pBaryonConstructor;
pBaryonConstructor.ConstructParticle(); ... }

void ExN04HadronPhysics::ConstructProcess() { ...
// Proton
pManager = G4Proton::Proton()->GetProcessManager();
// add process
theLEProtonModel = new G4LEProtonInelastic();
theProtonInelastic.RegisterMe(theLEProtonModel);
pManager->AddDiscreteProcess(&theProtonInelastic); ... }
```

Hadronic physics models

- There's a zoo ($\sim$ 10) of built-in hadronic models in G4 of various types:
  1. data-driven (MARS-type)
  2. paramterisation-driven (Geant 3 GHEISHA-type)
  3. theory-driven (DPMJET-type).

  model used so far in G4 is derived from the G3 GHEISHA model, to allow direct comparisons between MCs
- Code is open-source (but hard to understand!)
- Validation of the models is underway
- There is the possibility to run G4 with user-defined inclusive scattering cross-sections and final state production code.
- Example: provide formulas or tabulated data for cross-sections and final state production to treat interactions for a given particle, energy, material
- Implementation of user-defined models not done yet. This is the most important G4 aspect which remains to be tested (most difficult?). Getting help from G4 collaborators.

User actions and analysis

- User "hooks" at all levels of simulation process: beginning/end of runs, events, tracks, steps

- Basic MiniBooNE requirements:

1. Store track <u>and</u> parent track information in the same data structure
2. Speed up simulation as much as possible
3. Write ntuples on disk, possibly in a format which can be read-in by the redecay program, `BooBeamNT`, that is HBOOK ntuples

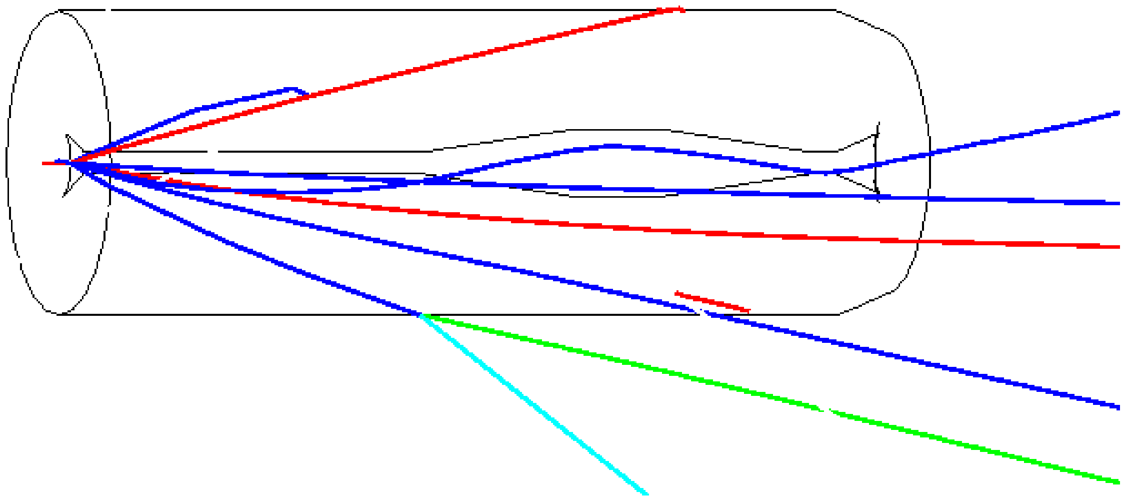
- Accomplishments so far:

1. Done. All the track and parent track information used by the beam MC is made persistent until the end of an event, and then stored
2. Little work done. Kill uninteresting particles based on their particle ID (gained factor of 12 in speed); haven't looked at all into energy cuts, yet
3. Some work done. The simulation writes Root trees, for the moment. Work will continue to manage to write HBOOK ntuples.

Customized tracking and visualization

- $p + \mathrm{Be} \longrightarrow \pi^+ + X$

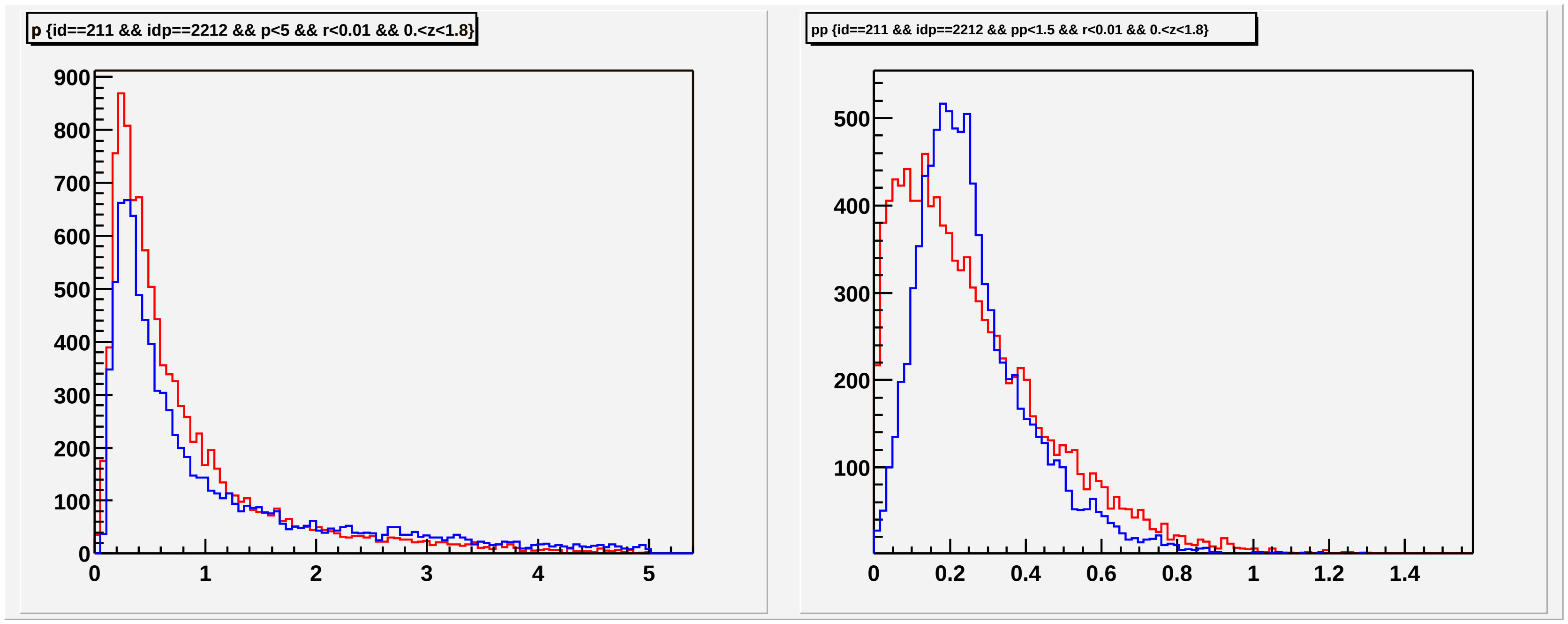
  $\hookrightarrow \mu^+ + \nu_\mu$



- Useful debugging tool to check geometry, magnetic field, particles and physics models constructed

Analysis: first plots and comparisons

- Just started to look at this
- Compare G4 (red) and G3 (blue) distributions of the total momentum p (left) and transverse momentum pp (right) for π^+'s created inside the target



- Comparison should involve: same (or similar) GHEISHA model, similar primary generator, and same geometry
  $\Rightarrow$ discrepancy not yet understood

To-do list

- Things we need a proof of principle of before we make a decision on G4:

  1. Implementation of user-defined cross-sections
     Work is in progress, with help of G4 collaborators
  2. Implementation of HBOOK interface
     This was easier with older versions of G4, more difficult but possible now

- Other important issues:

  1. Run G4 in MiniBooNE computing environment (ported this week by Chris)
  2. Put more realistic magnetic field and primary generator action
  3. Speed optimization
  4. Validation tests

Share responsibilities?

- Beam's group people need to join efforts soon on whatever beam MC we decide to adopt in the long-term
- I think it would be useful to have different persons responsible for different apsects of the beam MC, for example:
  1. Geometry and primary generator action
     study effect of modifying/removing/adding geometry volumes, put correct proton distributions, put correct magnetic field
  2. Physics models
     provide data and parametrisations for inclusive production cross-sections, by running various simulation models and by using data internal and external to MiniBooNE
  3. User actions and analysis interface
     responsible of general infrastructure of beam MC, memory management, analysis interface
- This sharing of responsibilities is particularly simple within G4, where different parts of code are independent from each other (no Fortran common blocks. . .)